IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

ATTY DOCKET NO.: AUS990918US1

In re Application of:	§	
	§	
DANIEL OTTO BECKER	§	Examiner: **NGUYEN, QUANG N.**
	§	
Serial No.: **09/583,519**	§	Art Unit: **2141**
	§	
Filed: **May 31, 2000**	§	
	§	
For: **DYNAMIC COMMAND SETS IN A COMPUTER MESSAGING SYSTEM IN A METHOD, SYSTEM AND PROGRAM**	§	

APPEAL BRIEF UNDER 37 C.F.R. 1.192

Mail Stop Briefs - Patents
Commissioner for Patents
Washington, D.C. 20231

Sir:

This Appeal Brief is submitted in triplicate in support of an Appeal of the Examiner's final rejection of Claims 1-5, 7, 9, 10, 12, 13, 15-20, 22 and 24 in the above-identified application. A one month extension of required time is required beyond the period for replied stated in the Advisory Action. That extension of time is requested by the attached Petition, and a check in the amount of $110.00 to cover the one-month extension of time is enclosed. Please charge the fee of $330.00 due under 37 C.F.R. § 1.17(c) for filing the brief, as well as any additional required fees, to IBM Deposit Account No. 09-0447.

Certificate of Transmission/Mailing

I hereby certify that this correspondence is being facsimile transmitted to the USPTO at 703-872-9306 or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450 on the date shown below.

Typed or Printed Name: Shenise Ramdeen *Date:* May 28, 2004 *Signature* Shenise Ramdeen

REAL PARTY IN INTEREST

The real party in interest in the present Appeal is International Business Machines Corporation, the Assignee of the present application as evidenced by the Assignment recorded at reel 010854 and frame 0860 *et. seq.*

RELATED APPEALS AND INTERFERENCES

There are no other appeals or interferences known to Appellant, the Appellant's legal representative, or assignee, which directly affect or would be directly affected by or have a bearing on the Board's decision in the pending appeal.

STATUS OF CLAIMS

Claims 1-5, 7, 9, 10, 12, 13, 15-20, 22 and 24 stand finally rejected by the Examiner as noted in the Advisory Action dated January 21, 2004.

STATUS OF AMENDMENTS

Appellants' Amendment B, filed on January 8, 2004, was entered by the Examiner, as noted in the Advisory Action.

SUMMARY OF THE INVENTION

Appellant's invention provides a method for enabling more efficient client-to-server messaging systems in computer networks that utilize command sets. The invention allows the client to initiate (activate or trigger) the execution of a function on the server by sending to the server (via the messaging system) a command that includes the exact name of the function (or function class). That is, the actual names of the executable files are included within the transmitted command to increase the likelihood that the function's executable file will be located and executed at the server (page 5, lines 13-19).

The commands are provided with similar names as the executable files (filenames) for the class file implementing the command. A command is passed to the server from the client with the class name, and the executable file (source code) having the same class name is retrieved and executed (i.e., the underlying operation is performed) on the network/server. When

the class name is not found in the library file names, a new class is created from related class code at runtime (page 5, lines 21-26).

The term "class file" or "class" as utilized within Applicant's invention refers to an executable function that is stored on the server with a specific class name similar to one of the names being utilized within commands issued by the client to the server. The use of the class file naming process operates similarly to a call to a subroutine or other remote functional call on the server. Thus, for example, once the class file is located on the server, an execution of the underlying function provided by the (source code of the) class file is triggered at the server (*see* page 8, lines 23-25 and line 29-page 9, line 3; page 11, lines 11-13). This is very different from prior art messaging systems (as implemented by *Tyra*) where providing the server with the file name results in initiation of a search for and/or retrieval of the file (or document) at the server and then transmitting the file back to the client.

Appellant's exemplary Claim 1 recites the following features/elements: (1) "receiving a command name at a server via said messaging system, said command name being sent by a client to **initiate** a particular one of multiple server **functions identified by an executable** having a name synonymous with said command name;" and (2) "dynamically **executing functions** on said server **associated** with said **class file**" (emphases added).

ISSUES

The primary issue for appeal is whether Examiner's rejection of Appellant's Claims 1-5, 7, 9, 10, 12, 13, 15-20, 22 and 24 under 35 U.S.C. § 102 as being anticipated by *Tyra, et al.* (U.S. Patent No. 6,442,565) is well founded. Tantamount to a resolution of that issue is a determination whether *Tyra* teaches each of the above features recited by Appellant's exemplary Claim 1 as well as each additional feature recited by Appellant's claims.

GROUPING OF THE CLAIMS

For purposes of this Appeal, all claims stand or fall together as a single group.

ARGUMENT

EXAMINER'S REJECTION OF APPELLANT'S EXEMPLARY CLAIM AS BEING ANTICIPATED BY *TYRA* IS NOT WELL FOUNDED AND SHOULD BE REVERSED.

Appellant hereby incorporates by reference the arguments proffered in Amendment B filed on January 8, 2004. Appellant's reiterate that *Tyra* does not anticipate Applicant's invention because *Tyra* does not teach each feature recited within Appellant's claims.

As stated within the Summary section, Appellant's exemplary claim recites the following features/elements: (1) "receiving a command name at a server via said messaging system, said command name being sent by a client to **initiate** a particular one of multiple server **functions identified by an executable** having a name synonymous with said command name;" and (2) "dynamically **executing functions** on said server **associated with said class file**" (emphases added). Several other claims of Appellant's invention provide this "executing" feature with respect to the located class file at the server. Claim 2 provides "executing said new instances of said class" and Claim 4 provides "instantiating functions of said class file on said server." Thus, for *Tyra* to anticipate Appellant's claims, *Tyra* must teach each of the above features (in addition to others of dependent claims not mentioned above). Tyra does not.

As stated within Amendment B and Amendment A (filed August 25, 2003), Appellant reiterates that *Tyra* is generally deficient as a 102 reference against Appellant's claimed invention because *Tyra's* transmission method does not execute functions of a class file at the server. Rather, Tyra's server operates solely as a repository of data, which transmits the data back to the requesting client and other systems on the network. Appellant's claimed invention, in contrast, transmits a command (from a client) that activates the execution of specific functions of a class file (identified by the class name) at the receiving node (e.g., server).

Tyra provides a "system for transmitting data content and performing operations on the data content" (Abstract). According to Tyra, the client system transmits a request for a particular operation and the server "constructs a response message," which is then transmitted to the client and other machines including a reference to the requested information (Abstract). The sections

of *Tyra* specifically cited by Examiner to support the 102 rejection does not teach each of the above features of Appellant's claim. Col. 2, lines 59-67, for example provides:

> A ... method...downloads files in a network along with changes in a format of files. A request for a file including an identifier for the file is received. It is determined whether the file exists in an associated memory, and a search is performed for the file by manipulating the identifier and comparing the manipulated identifier with available files. A located file and associated changes in a format of the file are downloaded based upon the determining and the searching.

Clearly, this section of *Tyra* merely describes a searching for a file by manipulating the file identifier and is devoid of any teaching of executing functions of a class file. Col. 17, lines 25-65 also provides a description of using a class name that is sent to a JVM to locate and return a corresponding file. Support for this analysis of *Tyra* is also found at col. 17, lines 44-46, which states that the "class" is loaded and returned to the application, and again at lines 61-63, which states that the class is loaded into memory and delivered to the application. Finally, even *Tyra's* title, abstract, and disclosure all support the argument that *Tyra's* system deals specifically with "transmitting data content" to a requesting client seeking updates to the data.

Tyra is clearly devoid of any teaching of "dynamically executing functions ... associated with said class file." The "executing" being described by Appellant's claims refer specifically to actual implementation of the function performed as a characteristic of the class file and NOT operations by the server to locate the class file (as is suggested by Examiner and described by *Tyra*). As described in the cited sections and explained above, *Tyra* merely returns the requested data/file to the client system. *Tyra* does not provide an associated functional execution on the server in response to locating the file on the server/network. Appellant's invention, in contrast, relates specifically to triggering execution of a network-level functionality/application whenever the class name within a command locates the corresponding executable with similar file name.

At page 3 of the Final Action, Examiner's provides several conclusory arguments regarding the correct attributes/meaning of the "executing" feature as one related to functions

performed by the server to locate and manipulate the class file. Examiner also reiterates these arguments in the Advisory Action.

Examiner's comments that Appellant's "executing" is somehow synonymous with functions required to locate/find the file ("such as searching... comparing the manipulated the identifier, and loading and transmitting the file... to the client") is flawed. Execution of an underlying programmed function of a class file is inherently different from a combination of all the described server functions associated with the file (including searching for and manipulating/updating the class file). Executing functions refers specifically to executing the source code associated with the class file and NOT performing the server operations of locating and/or manipulating the class file. In computer programming terminology and program code execution environment, an actual function performed by a program component is different from operations performed on the component by another executable function (e.g., server operations).

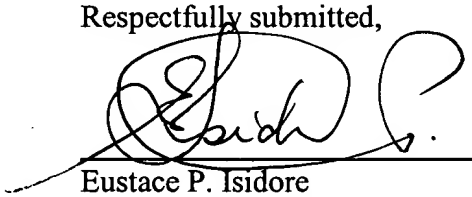
Further, Appellant actually differentiates the term "executing" from "searching" "manipulating" and "loading", etc., by using each gerund, where provided within the claims and specification, in its respective context. Examiner cannot therefore assume that a very specific reference to an "executing" feature is the same as another, different feature (e.g., "searching" or "loading") when the different features are used separate from and in a clearly distinct manner from the "executing" feature of the claim.

From the above arguments, it is clear that while *Tyra* is a file retrieval mechanism that searches for and returns the identified class file to the client, Appellant's claimed invention provides a remote call feature whereby functions of the executable are initiated at the server by including the executable class/filename within the command sent via the messaging system. Thus, Appellant has clearly shown that *Tyra* does not teach the various features that are recited by Appellant's exemplary claim. Appellant's claims are therefore not anticipated by *Tyra* and should be allowed. For these reasons, Examiner's rejection of Appellant's claims is not well founded and should be reversed.

CONCLUSION

Appellant has pointed out with specificity the manifest error in the Examiner's rejections, and the claim language which renders the invention patentable over the combination of references. Appellant, therefore, respectfully requests that this case be remanded to the Examiner with instructions to issue a Notice of Allowance with respect to all pending claims.

Respectfully submitted,



Eustace P. Isidore
Registered with Limited Recognition (see attached)
Dillon & Yudell LLP
8911 North Capital of Texas Highway
Suite 2110
Austin, Texas 78759
512.343.6116

ATTORNEY FOR APPELLANTS

APPENDIX

1. A method for generating a command in a computer messaging system, comprising the steps of:

receiving a command name at a server via said messaging system, said command name being sent by a client to initiate a particular one of multiple server functions identified by an executable having a name synonymous with said command name;

utilizing said command name to automatically load a class file having a name including said command name; and

dynamically executing functions on said server associated with said class file.

2. The method of claim 1, further comprising:

comparing said command name to existing classes on said server;

when said command name does not match a name of an existing class, creating a new instance of said class; and

executing said new instance of said class.

3. The method of claim 1, further comprising:

deleting a current command at runtime.

4. The method of claim 1, further comprising:

adding a new command at runtime; and

when a class file associated with said new command is available at a source other than said server, which is accessible from said server, dynamically loading said class file from said source and instantiating functions of said class file on said server.

5. In a computer system connected to at least one other network computer system, a method for employing a command in a messaging system operating on said computer system, said method comprising the steps of:

selecting a command desired to be executed at the network computer system connected to said computer system, wherein said network computer system comprises a set of class files that carry out specific functions when initiated; and

transmitting, within a message to the network system in which the selected command is to be executed, a command name for the selected command which matches a portion of a class filename for a class implementing the selected command, wherein said selected command triggers an activation and execution at said network system of functions associated with said class;

comparing said command name to existing classes on said messaging system;

when said command name does not match any one of said existing classes, creating a new instance of said class; and

executing said new instance of said class.

6. (withdrawn)

7. A method for employing a command in a computer network, said method comprising:

receiving a message containing a command name; and

comparing said command name to existing said class files;

locating a class file having an executable class filename that is substantially similar to said command name;

loading and instantiating said class file and functions provided thereby; and

when said command name does not match any one of sad existing classes, executing a new instance of said class.

8. (canceled)

9. (currently amended) In a computer system connected to a network computer, a system for generating a command response, comprising:

selection means for selecting a command to be executed within a computer network;

transmission means for sending a command name within a message to the network computer, such that said command name causes and execution of said selected command, which matches a portion of a class filename for a class implementing said selected command;

means for adding a new command at runtime; and

means, when a class file associated with said new command is available at a source other than said server, which is accessible from said server, for dynamically loading said class file from said source and instantiating functions of said class file on said server.

10. The system of claim 9, further comprising:

means for comparing said command name to existing classes;

means for loading said class file having said command name;

means for creating a new instance of said class when said class cannot be loaded by said system; and

execution means for executing said new instance of said class.

11. (canceled)

12. The system of claim 9, further comprising:

means for deleting a current command at runtime.

13. A system for employing a command via a messaging system, comprising:

means for receiving a command name into a server via said messaging system;

means for comparing said command name to existing classes;

means for utilizing said command to automatically load a class file having a name including said command name; means for dynamically executing functions associated with said class file after said class file is loaded;

means, when said command name does not match any of said existing classes, for creating a new instance of said class; and

execution means for executing said new instance of said class.

14. (withdrawn)

15. The system of claim 13, further comprising:

means for adding a new command at runtime; and

means, when a class file associated with said new command is available at a source other than said server, which is accessible from said server, for dynamically loading said class file from said source and instantiating functions of said class file on said server.

16. The system of claim 13, further comprising:

means for deleting a current command at runtime.

17. A computer program product within a computer readable medium having instructions for generating a command, comprising:

instructions within said computer program product for receiving a command name from a client computer via a messaging system;

instructions within said computer program product for utilizing said command name to automatically load a class file having a name including said command name; and

instructions within said computer program product for dynamically executing functions associated with said class file after said class file is loaded.

18. The computer program product of claim 17, further comprising:

instructions within said computer program product for comparing said command name to existing classes on said server;

instructions within said computer program product for creating a new instance of said class when said class name does not match any one of said existing classes; and

instructions within said computer program product for executing said new instance of said class.

19. The computer program product of claim 17, further comprising:

instructions within said computer program product for adding a new command at runtime; and

instructions within said computer program product for dynamically loading said class file from said source and instantiating functions of said class file on said system when a class file associated with said new command is available at a source other than said system.

20. The computer program product of claim 17, further comprising:

instructions within said computer program product for deleting a current command at runtime.

21. (withdrawn)

22. A computer program product within a computer readable medium having instructions for employing a command in a messaging system, comprising:

instructions within said computer program product for selecting a command to be executed;

instructions within said computer program product for transmitting, within a message to a system in which the selected command is to be executed, a command name for the selected command which matches a portion of a class filename for a class implementing the selected command, wherein said command name triggers an execution of a related function of said class when received at a recipient computer system;

instructions within said computer program product for comparing said command name to existing classes on said server;

instructions within said computer program product for loading said class file having said command name;

instructions within said computer program product for creating a new instance of said class; and

instructions within said computer program product for executing said new instance of said class.

23. (withdrawn)

24. The computer program product of claim 22, further comprising:

instructions within said computer program product for deleting a current command at runtime.